

DOCUMENT RESUME

ED 048 777

24

EM 008 851

TITLE CITY III Operator's Manual.
INSTITUTION Envirometrics, Inc., Washington, D.C.
SPONS AGENCY Office of Education (DHEW), Washington, D.C. Bureau
of Research.
BUREAU NO BR-9-0515
PUB DATE Oct 70
CONTRACT OEC-0-9-180515-4530(010)
NOTE 21p.; See also EM 008 848, EM 008 849, EM 008 850,
EM 008 852

EDRS PRICE EDRS Price MF-\$0.65 HC-\$3.29
DESCRIPTORS City Planning, Computer Assisted Instruction,
*Computer Programs, Educational Games, *Input
Output, Management Games, *Manuals, Metropolitan
Areas, *Simulation, *Urban Areas, Urban Environment
IDENTIFIERS *CITY III

ABSTRACT

CITY III is a computer-assisted simulation game of an urban system involving player operation of and interaction with economic, social, and government components. The role of operator in the game is to take the handwritten inputs (decisions) from the CITY III participants, process them, and return output which initiates the next round of decision-making. The CITY III package consists of two seven-track tapes, the master or program tape, and the data tape. The program was written for the Univac 1108, executive system eight. Input can be processed either by means of punched cards or a terminal input-output facility such as a teletype, but the output must be printed on a 132-character hardware printer. The procedures involved in obtaining a new round of output are described. Thirty-two job control statements are defined which, in prescribed combinations, can perform any of the operations in the CITY III repertory. Examples illustrate the sequence in which job control statements must be processed in order to execute all the programs in various combinations. The structure of the CITY III data tape is described.
(Author/JY)

ED048777

U.S. DEPARTMENT OF HEALTH, EDUCATION
& WELFARE

OFFICE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED
EXACTLY AS RECEIVED FROM THE PERSON OR
ORGANIZATION ORIGINATING IT. POINTS OF
VIEW OR OPINIONS STATED DO NOT NECES-
SARILY REPRESENT OFFICIAL OFFICE OF EDU-
CATION POSITION OR POLICY.

CITY III

OPERATOR'S MANUAL

ENVIROMETRICS, INC.

October, 1970

EM 008 851

INTRODUCTION

CITY III is a computer-assisted game of an urban system involving player operation of, and interaction with, economic, social and government components.

Players make decisions based on their interpretations of one year's activity as described by the computer output. The computer processes their decisions, simulates the next year's activity, and produces new player output. Thus, this operator's manual is provided because players and users must work with the computer.

This manual is neither a beginner's guide to a Univac 1108 system nor a highly technical documentation of the CITY III programs and data. Since it lies somewhere between the two cases, the manual may confound the novice and offend the expert; please be neither. This manual will be of use to anyone who is responsible for taking the handwritten inputs from the CITY III participants and, in due time, returning the next round of output to them.

Section A of the manual provides an overview of the programs. Section B, Description of Operations, illustrates when and why each program or combination of programs should be used. Section C explains in simplified fashion the meaning of each of the thirty-two (32) different job control statements which in prescribed combinations can perform any of the operations which the CITY III package offers. Section D lists nine examples of the operations along with the required sequence of the job control statements. The Appendix explains the structure of the data tape.

Director and Operator Defined

Let us first distinguish the operator from the director. Theoretically, the director is responsible for all activity within the game-room; whereas, the operator oversees the connection between the game-room and the computer.

A Word About the Manual and CITY III

Your understanding of this manual will be enhanced by first reading the Director's Guide*, which is referenced twice in this text. Furthermore, your comprehension of the game concepts would make clear the role of the EDIT program and its relationships with the simulation (the CITY3** program). Either reading the Players' Manual or observing a CITY III play could aid in the latter case.

An individual with a computer science background who in particular understands the Univac 1108 system will quickly digest the specifications provided in Section A of the manual. That information is mostly necessary accounting data for the staff of the computer installation. For example, the "run" card which identifies the user also includes indicators of the system output requirements such as the upper limit of pages of output and maximum execution time. To satisfy these perfunctory needs both the novice and the expert must examine the first two pages.

While the expert may not need Sections C and D as he can determine the requisite job control merely through verbal explanation of the programs, the novice may need to rely heavily on these two sections. First, he should match the operation(s) he wants performed with those listed in Section D. Then he should order the job control cards, listed in Section C, as prescribed in the space opposite the chosen operation description in Section D.

Once he submits all the cards in the proper order as outlined in the Director's Guide the job can be executed and the output returned. If there is some error in the statement due to a peculiarity of the particular computer system, the CITY III operator can explain the intention of the erroneous statement to a staff member of the computer installation. Without necessarily understanding the action explanation, the novice can refer to the column in Section C opposite the particular job control statement. From that information the staff member should be able to make the proper adjustment.

* The text minus the appendices and introduction would be most helpful.

** Throughout this manual CITY3 refers to the computer program which executes the simulation and CITY III refers to the entire game.

All operators should keep a log of the data so that they know the precise contents and location of each file or a round's data base. Figure 1 of Appendix A is an illustration of this procedure.

You should understand the use of the term "round" and how rounds are numbered. The director/operator must deal with two rounds: the player's output is known as a "round," and the data base stored in either a tape file, or as it remains in the computer system, is a "round" of data. Players make decisions from a data base numbered the same as the round of output with which they are playing.

Thus, care should be taken in discussing "rounds" with the players, computer installation staff, or the director to avoid misunderstandings.

A Round 0 data base is only produced by the execution of the LOAD program. LOAD produces no round of output. Consequently, once LOAD has generated an initial data base, a subsequent round of output along with a subsequent round's data base (both of which have the same round number) can only be produced through the execution of the simulation (CITY3) on the previous round's data base (perhaps altered by EDIT). Thus, only execution of CITY3 produces a new round of output and a new round's data base, both numbered the same. An altered round is the data base generated by the execution of the EDIT (input processing) program to the players' round decisions. That data base retains its round number; e.g., EDIT executed on the Round 1 data base generates an altered Round 1 data base.

A. CITY III PROGRAM

The CITY III package consists of two seven-track tapes, a master or program tape and a data tape. The program requires up to 38K of central processing unit (CPU) core. The CPU running time varies from just over two minutes to nearly twenty minutes per round depending on the particular city being played (see Figure 1a on page 2 of the Director's Guide for tabular presentation of basic parameters for each city). The number of pages of output per round also varies according to the city being played; usually there are between 150 and 300 pages.

The program was written for the Univac 1108, executive system eight; the routines were coded primarily in Fortran IV (about one tenth of the programs are written in 1108 assembly language). Input can be processed either by means of punched cards (and card reader) or a terminal input/output facility such as a teletype. The output, however, must be printed on a 132 character hardware printer.

The program tape or master tape contains four operating programs (LOAD, EDIT, CITY3, and MAPS) along with the symbolic representation (card images) of the load decks for each of the five cities. The load decks are the punched cards containing all the information necessary to define each city.

The following table briefly describes the operation performed by each program.

<u>PROGRAM NAME</u>	<u>DESCRIPTION OF OPERATIONS</u>
LOAD	Converts chosen load deck data into data base appropriate for the execution of the game, i.e., it enables the operator to run CITY3 off the Round 0 base and thus obtain Round 1 output.
EDIT	Records input decisions which change the current round's (most recent round) data base; identifies errors; checks that decisions are legitimate and follow proper format.
CITY3	Runs the simulation; using the previous round's data base unchanged or altered by EDIT, CITY3 produces the new round of output and the current updated data base.

MAPS

Produces an extra copy of the status maps:
(Economic, Government, Preempt, Demographic,
and Social Decision-makers).

To execute any or all of these programs, the operator must copy the whole master tape into the system's core. Note that this tape is used for system input only, whereas the data tape is used for both input (the previous data base) and output (the new data base). Consequently, after the contents of the master tape have been copied onto the temporary program file (the system's core), the master tape may be released or freed.

The data tape contains rounds of data bases for any of the five CITY III configurations. Each round is marked or "tabbed" so that any particular data base can be accessed at any time.¹⁾ The tape we have provided can hold up to 100 rounds of data. Initially, there are only two data bases stored on the data tape; their generation is explained later. (See page 17)

¹⁾ The description of the data tape structure is Appendix A.

B. DESCRIPTION OF OPERATIONS

The players' mechanical tasks are very limited. Presented with a round of output, they must make decisions based both on that output and game-room-generated information. The decisions must be coded properly onto input forms. It is the director's task to process the inputs from the completed forms through the proper execution of programs culminating in the updated round of output (new round) for the players to resume decision-making.

The procedures involved in obtaining a new round of output are defined as operations. The following simplified sequence indicates the elements of the operations and thus outlines the operator's role. [Note that the director and operator may be the same person or group of people.]

Sequence of Operator Procedure

1. Ready director and player inputs
for execution
(Cards punched and ordered)
2. Select data base on
which program(s)
to be executed.
3. Determine programs
desired for execution
(normally EDIT followed by CITY3)
4. Determine file position(s) (if any)
into which new or altered data base(s)
are to be stored. (See Appendix A)
5. Set up Job Control Cards
(Remember to place all inputs in proper sequence
immediately after the "@XQT EDIT" statement)
6. Submit program to operator of
computer installation
7. Receive output and cards from processing staff
at completion of run

The director may then distribute the new output to the players.

The following describes the sequence of operator procedures with each section focusing, but not exclusively, on one of the programs.

A word about the first round of output. It is the only round for which the players cannot have made decisions. This round, like all others, is produced by executing the simulation (CITY3) on the previous round's data base, possibly altered by the EDIT (input processing) program. In this instance, the "previous round's data base" is the Round 0 or starting data base; Round 0 data bases are only generated by the execution of the LOAD program.

LOAD Program

When the director determines that a play will begin with a new city, the new Round 0 base must be generated or else have been generated and saved in a data file at an earlier time. Recall that the basic information which defines each of the city configurations (the load deck) is contained on the program tape in the form of card images. LOAD simply translates these card images into a data base upon which the other three programs (EDIT, CITY3, MAPS) may be executed.

Thus, in terms of the schema listed above, step 1 is ignored. In step 2 the director selects the city to be played based on an approach outlined in the first chapter of the Director's Guide. LOAD must be executed (step 3). The file location may be the first place on a tape -- one especially for rounds of that particular city -- or on any available position on a "used" data tape.

Procedures involved in step 5 are described in great detail later in this manual. By the time this step is reached, the operator should know exactly what is required and can select the sequence of cards described as an operation #1 on page 15. Steps 6 and 7 are self-explanatory.

EDIT Program

Once the players and/or director have coded decisions based on a certain output (data base), the EDIT program must be executed to record these changes on the designated data base. EDIT also rejects invalid decisions and provides the explanation for each rejection.

EDIT may be executed on a Round 0 data base as generated by LOAD. Only director inputs (see Director's Guide) may be included in this execution. Thereafter, the input forms submitted to the operator may include both player and director decisions.

Step 1 notes the requirements that all inputs must be properly keypunched and ordered. Although this is listed as an operator's assignment, it need not be done by him. In fact, the instructions for completing this step are provided in the Director's Guide. It may prove advantageous to thoroughly proofread the input forms prior to the keypunching operation.

The remaining steps apply most directly to the execution of EDIT. Most frequently, the EDIT program will be executed immediately preceding the execution of the simulation (CITY3). This circumstance is described under the CITY3 explanation below.

EDIT has access to the entire round's data base. Sequentially and singly the inputs are first printed (with all blanks removed) checked for validity, and then either rejected (with the reason for rejection printed out) or accepted and the decision is recorded. At the end of processing of inputs, EDIT completes land auction, determines availability of funds for federal-state aid, and accepts or rejects all attempts to redistrict school and municipal services. Once the results of these three actions are recorded in the data base and printed, EDIT terminates.

CITY3 Program

CITY3, the simulation of a year's activity, is executed on any round's data base (normally, although not necessarily, altered by EDIT) to produce an updated, or new round. In addition to this data product, CITY3 prints the total output on which the players and director base their decisions and analyses.

Frequently CITY3 is executed along with EDIT. The edits (the output from EDIT) and the simulation output (from CITY3) are generally considered as a unit. Therefore, this operation, which makes more efficient use of the machine time, specifies the execution of CITY3 immediately after EDIT (i.e., in the same computer job). Insertion of an additional job control statement enables the altered data base (after EDIT and before CITY3) to be saved in a file. The data base generated by the execution of CITY3 should always be saved.

Job Control Operation numbers 7 and 8 on page 16 illustrate the two standard procedures for producing the new round. There is a circumstance under which the operator may want to execute EDIT and CITY3 in separate jobs: if one chooses to examine the edits prior to running the simulation off the (thus) altered base. In the case that the edits showed some errors on inputs which were considered vital to the play (i.e., the results of these decisions were deemed necessary for the next round of output), the operator or director could correct the errors and rerun EDIT, thereby avoiding the wasted time and expense of producing unusable new output.

MAPS Program

Up to sixteen additional copies of the status maps (economic, demographic, government, public institutional and social decision-makers) can be obtained by executing MAPS on any round's data base. MAPS may be executed on one of two data bases which reflect the various phases of the status of any one particular round. For example, if MAPS was executed on the data base produced by the simulation, all maintenance and newly inputted construction projects would appear completed. However, if MAPS was executed after EDIT but before CITY3, depreciation and the results of migration will not be represented. These two options for MAPS differ from the status maps printed by the simulation program (CITY3) which do show plant depreciation but not plant maintenance. As there is no way to duplicate these CITY3-generated maps exactly the director must decide which option more closely reflects the status appearing on the players' output.

The following table is provided to enable the director and/or operator to determine the temporal factors involved in the two MAPS options as well as the simulation-produced maps. It includes only the factors which may vary. The players' output is assumed to be for Round T; thus the previous round's data base is indicated by T - (minus) 1.

The columns A, B and C represent the status map factors produced by: (A) the execution of the simulation (CITY3) on the previous round's data base (normally altered by EDIT); (B) the execution of MAPS after EDIT and before CITY3 on the altered Round T-1 data base, and (C) the execution of MAPS after CITY3 (in this case MAPS operates on the unaltered Round T data base). Note that if there are no inputs for any round, CITY3 may be run anyway to produce new output and data. MAPS executed prior to this simulation reflects the same factors' status as case B.

FACTORS	A	B	C
SOCIAL DECISION MAKER CONTROL FOR ROUND:	T	T	T+1
ALL LAND USES (EXCEPT UT AND RAIL) CONSTRUCTION PROJECTS COMPLETION STATUS THRU ROUND:	T	T	T+1
UT AND RAIL CONSTRUCTION PROJECTS COMPLETION STATUS THRU ROUND:	T	T	T
LAND USE DEMOLITION PROJECTS COMPLETED THRU ROUND:	T	T	T
LAND USE VALUE RATIOS AFTER DEPRECIATION FOR ROUND:	T	T-1*	T*
LAND USE VALUE RATIOS AFTER MAINTENANCE FOR ROUND:	T-1**	T-1	T
MIGRATION FOR ROUND:	T	T-1	T
PROPERTY OWNERSHIP FOR ROUND:	T	T	T
UTILITY SERVICE FOR ROUND:	T	T	T
ZONING FOR ROUND:	T	T	T

* The effects due only to depreciation can only be seen on the Demographic Status Map generated by the execution of CITY3 (column A). Columns B and C refer to value ratios which have been "maintained." Only where there has been no maintenance will the value ratios apparent in these last two cases be equivalent to the value ratios due only to depreciation.

**Like depreciation, maintenance is done during the execution of the CITY3 program. As is noted in Chapter 2 of the Director's Guide, maintenance is completed after depreciation, hence the discrepancy in column A for this factor and the preceding one. For the sequence of the routines in the CITY3 programs, see Section B of Chapter II of the Director's Guide.

Based on the above considerations, the director can complete step 2 in the operations sequence. MAPS must be executed (step 3). Ignore step 4. Operations number 4 and 5 on page 15, and 9 on page 16, illustrate ways to organize the job control for three options of MAPS execution (step 5). Steps number 6 and 7 are the same as for the other programs.

C. JOB CONTROL EXPLANATION

To supplement the basic rationale behind the job control required to properly execute each program above, the following list of the job control statements is paired with a brief explanation of the action taken when each statement is encountered by the computer. The statements appear as they should be keypunched; the "@" character (7-8 punch) should be in column 1 of each statement. A lower case letter indicates that some number or letter(s) should be keypunched at that location on the cards.

<u>Number</u> ^{2]}	<u>Job Control Statement</u>	<u>Action</u>
1.	@RUN CITY3,a,b,c,d	The computer installation's accounting information is provided by columns a, b, c, and d where: a. is the account number (6 characters) assigned by the installation; b. is the project name of 0-12 characters selected by the user; c. is the maximum run time in minutes; d. is the maximum pages.
2.	@ASG,T TPF\$,F/64/TRK/1000	This statement assigns a name (TPF\$) to the temporary file available for the program while the system is being used for the CITY III job.
3.	@ASG,T GM	This assigns a temporary file name and storage location in the system's file. GM is used for the data base storage and manipulation during the execution of the job.

^{2]}The number and period are for reference purposes and should not be keypunched.

- | | |
|--|--|
| <p>4. @ASG,T TAPE,8C,xW
 5. @ASG,T RND,8C,yW
 6. @ASG,T NEW,8C,zW</p> | <p>Each assigns a temporary reference name to the tape numbered x, y, or z (e.g., tape number y would hereafter be referred to as RND); TAPE is the program tape while RND and NEW are data tapes (this package consists only of TAPE and RND). When these tape statements are encountered by the computer the mounting of a specific tape reel is demanded. "W" denotes that the data tapes are to be "ringed."</p> |
| <p>7. @ASG,A MAPSF</p> | <p>Assigns a temporary name to the catalogued file in the system's disk drum. MAPSF will only be used if the printout images are to be saved so that later it can be provided as output.</p> |
| <p>8. @DELETE,C MAPSF</p> | <p>Deletes any catalogue file referenced as MAPSF.</p> |
| <p>9. @CAT,P MAPSF,F/64/TRK/1000</p> | <p>Requests and obtains system drum storage as catalogued file with name MAPSF.</p> |
| <p>10. @FREE TPF\$
 11. @FREE TAPE
 12. @FREE RND
 13. @FREE MAPSF</p> | <p>"Free" statements tell the operator of the machine that the job no longer needs the tape specified as the argument of the statement. It is a courtesy measure in most instances. In effect it negates the action taken by an assign statement (@ASG)</p> |
| <p>3]14. @MOVE RND.,x</p> | <p>This statement positions the tape RND (the data tape) to the beginning of any specific file (data base) on that tape. Recall that each data base is marked or "tabbed." This moves the tape to a particular mark, identified by a non-zero digit indicated by the x.</p> |

^{3]} See explanation of tape positioning in Appendix A.

15. @COPIN TAPE
- This statement copies the entire contents of the program file located on the tape identified as TAPE onto the system drum. This permits execution of any of the programs and access to the symbolics of the load decks.
- 3] 16. @COPY,G RND.,GM.
- This copies the contents of the file starting with the point at which the tape drive is positioned (at the beginning, if no @MOVE statement precedes this one) -- i.e., on data base from the tape RND -- onto the system's GM file.
- 3] 17. @COPY,GM GM.,RND.
3] 18. @COPY,GM GM.,NEW.
- The data base as altered in the GM file is copied out onto either data tape, RND or NEW. This file is marked on the data tape so it can be accessed via an appropriate "move" at a later time.
- 3] 19. @XQT LOAD
4] 20. @XQT EDIT
21. @XQT CITY3
22. @XQT MAPS
- These initiate execution of the programs. Note that all of them require the assignment of a GM file.
23. @ADD DUNBEATH
24. @ADD METRO
25. @ADD MORAY
26. @ADD TRICITY
27. @ADD LOTHIAN
- One of these statements must follow an @XQT LOAD statement; this supplies the data on the starting configurations which LOAD then translates into a Round 0 data base.
28. @BRKPT PRINT\$/MAPSF
- This directs any output from the subsequent execution of one of the four programs to be stored on the drum catalogued file MAPSF. This enables multiple and/or later printing of the output. Basically this means that one is printing onto a file rather than onto paper.

3] See explanation of tape positioning in Appendix A.

4] All input decisions must be placed immediately after this statement.

29. @BRKPT PRINT\$ This closes whatever PRINT\$ file (which is only for output) being used at this time. In the case in which this package requires the statement, it terminates the saving of output on the cataglogued file MAPSF.
30. @SYM,SU MAPSF,,PRn Whatever output that was stored in MAPSF is now "symmed" to the printer number "n." This produces the standard computer output. This statement may be repeated up to 16 times.
31. @REWIND RND. Rewinds the data tape so the drive points to the first file. It enables "back-spacing."
32. @FIN Terminates the run. If no printing instructions preceded this, the output will now be able to be printed.

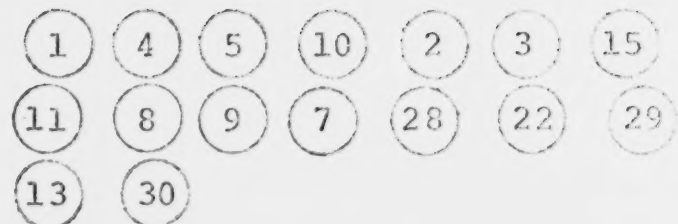
D. PRECISE JOB CONTROL FOR VARIOUS OPERATIONS

The following table gives the reference numbers of the job control statements in the sequence in which they must be processed in order to execute all of the programs in various combinations. The data tape named RND is assumed to have two files: the first file is Round 0; the second is Round 1 as generated by the execution of the simulation on the Round 0 data base (these two operations are examples 1 and 2 below). The tapes named TAPE, RND and NEW correspond to the program tape, the enclosed data tape, and a new tape for data which the user can supply.

OPERATION	TAPES REQUIRED (BY NAME)	JOB CONTROL STATEMENTS IN SEQUENCE
1. Execute load to get initial data base (round 0) for any city.	TAPE, RND	1 4 5 10 2 3 15 11 19 either 23 24 25 26 or 27 17 32
2. Execute CITY3 on the Round 0 data base	TAPE, RND	1 4 5 10 2 3 15 11 16 21 17 32
3. Execute EDIT on the Round 1 data base (assume for statement #14 that $x = 1$)	TAPE, RND	1 4 5 10 2 3 15 11 14 16 20 (immediately followed by all input decisions) 17 32
4. Execute MAPS on the Round 0 data base	TAPE, RND	1 4 5 10 2 3 15 11 16 12 22 32
5. Execute load, EDIT on Round 0 base (director inputs) MAPS, and CITY3. Save only the Round -1 data base and copy out the first file of RND	TAPE, RND	1 4 5 10 2 3 15 11 19 either 23 24 25 26 or 27 20 (immediately followed by all input decisions) 22 21 17 32

6. Print multiple copies of maps of Round 0 data base

TAPE, RND



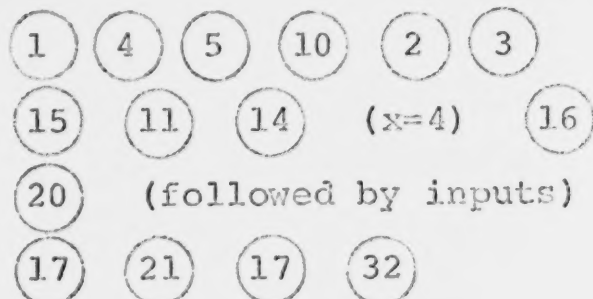
(repeated up to 16 times)



For the following examples, assume the data tape structure given in the figure on page 18.

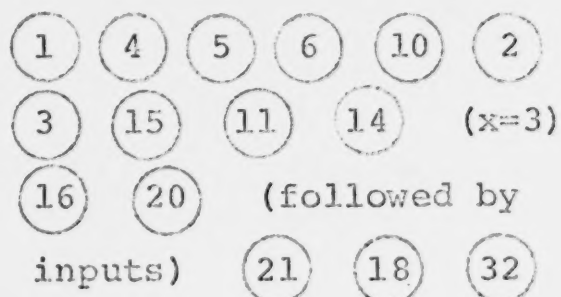
7. Execute EDIT on the Round 3 base, copy out that base, and then execute CITY3 on this newly altered Round 3 data; copy out this base.

TAPE, RND



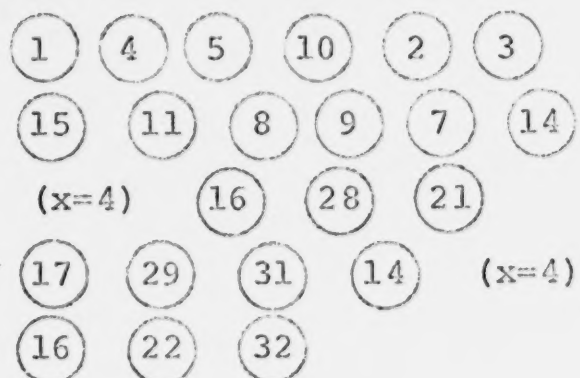
8. Execute EDIT on Round 2 base and then execute CITY3 on the altered base. Save only the Round 3 data base on a new data tape (NEW).

TAPE
RND
NEW

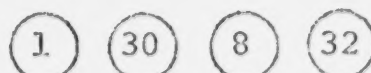


9. Execute CITY3 on the Round 3 data base and save that output on a catalogued drum file. Save the Round 4 data base in the next file on tape RND. Print one copy of MAPS from the Round 3 base.

TAPE, RND



10. Print the output stored in the computer by one of the steps in operation #9. Terminate the storage of that output.



As an illustration, the properly keypunched and arranged job control statements for operation #7 are:

	<u>CARD</u>	<u>IMAGE</u>
(1)	@RUN	CITY3,123456,SAMPLE,5,300
(10)	@FREE	TPF\$
(2)	@ASG,T	TPF\$,F/64/TRK/1000
(3)	@ASG,T	GM
(4)	@ASG,T	TAPE,8C,169
(5)	@ASG,T	RND,8C,170
(15)	@COPIN	TAPE
(11)	@FREE	TAPE
(14)	@MOVE	RND.,4
(16)	@COPY,G	RND.,GM.
(20)	@XQT	EDIT

Inputs such as \$CASH/=OU/C,15000,MS1,PVT,CUR etc.

(17)	@COPY,GM	GM.,RND.
(21)	@XQT	CITY3
(17)	@COPY,GM	GM.,RND.
(32)	@FIN	

The data tape enclosed in this package has the Round 0 data base of DUNBEATH in the first file. CITY3 was then executed on the Round 0 base to provide the accompanying sample output. The Round 1 data base is stored in the second file.

APPENDIX A

STURCTURE AND MANIPULATION OF A CITY3 DATA TAPE

TAPE		R N D
Mark #	File Contents	File #
	Round 0 data base - result of LOAD on card images stored on the Program Tape	1
1	Round 1 data base - result of simula- tion (CITY3) on Round 0 data base	2
2	Round 1 data base after EDIT on Round 1 data base.	3
3	Round 2 data base after simulation based on Round 1 EDIT	4
4	Round 3 data base after simulation based on Round 2 (no EDIT)	5
5	E M P T Y	

When a data tape is mounted on a computer disk drive the first block of information that can be accessed is the file of data which begins at the start of a tape. Each file has its beginning and end points identified by a mark; the beginning of the second file is equivalent to the end of the first file. This particular spot is the location of the first mark (see above figure).

By using the @MOVE statement, the beginning of any file on a data tape may be reached. Once that is accomplished, when the job requests that a data file (data base) be read into the computer core so that it may be operated on, the file beginning with the mark currently pointed to is read.

After the file read is completed, the tape drive points at the next mark. This feature prevents any of the programs from copying over, and thus losing, the previous contents of that file. If it is desired to copy over or access a data file which precedes the mark on the tape to which the tape drive is pointing, an @REWIND (#31) statement must be used; once that action is completed, the proper move can be made.

The above figure illustrates a data tape construct. In this case there are five files (five different data bases), the generation of which is started with the file contents. The marks begin with the second file; thus, to access any data base, one must move to the mark which is numbered one less than the number of the file. Any attempt to move to the fifth mark (the sixth file) on this tape would be useless as there are only five files.

Note that all of the data bases were produced sequentially for one city configuration. Assuming that the programs producing these files were all run on the same computer job, the order of execution was: LOAD, CITY3, EDIT, CITY3, and CITY3. If the director erred in producing the Round 3 data (file #5) without inputs, he could submit the inputs and execute EDIT and CITY3 off the Round 2 data base to produce the Round 3 data base, and output. After moving to the third mark to copy into the system (onto the GM file) the Round 2 data base (file #4), he should execute EDIT. If he desired to save the thus modified Round 2 data base he could then copy out the data onto the file which contains the incorrect Round 2 data (this would require no move since after the read the tape drive is pointing to mark #4, file #5). This again moves the tape drive pointer.

Execution of the simulation on the altered Round 2 (which is still in the system) then produces the Round 3 output and data base in the system's GM file. Another copy out fills a sixth file.